Lab2

2023-10-30

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

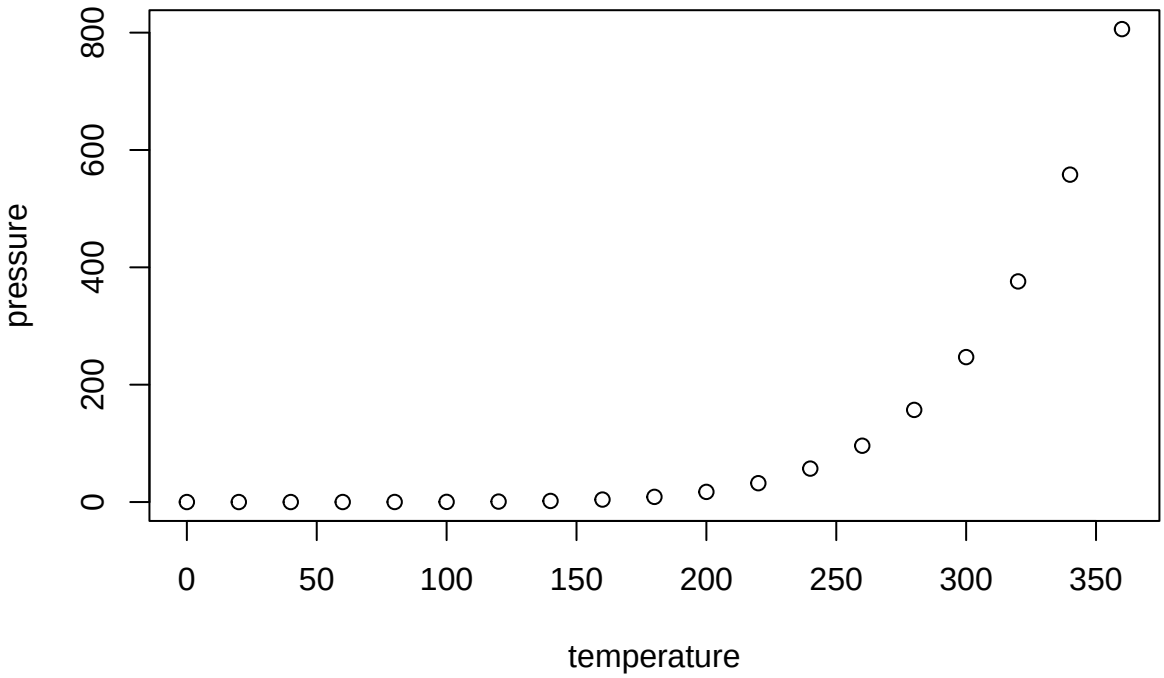
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```r
library(ggplot2)
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2movies)

data(movies)

#1
oldest_movie_year <- min(movies$year, na.rm = TRUE)
most_recent_movie_year <- max(movies$year, na.rm = TRUE)

cat("The oldest movie was produced in", oldest_movie_year, "\n")
```

```
## The oldest movie was produced in 1893
```

```r
cat("The most recent movie was produed in", most_recent_movie_year, "\n")
```

```
## The most recent movie was produed in 2005
```

```r
#2

# check the proprtion of movies with budget included in the data
budget_included <- sum(!is.na(movies$budget))
budget_not_included <- sum(is.na(movies$budget))
budget_included_proportion <- budget_included / nrow(movies)
budget_not_included_proportion <- budget_not_included / nrow(movies)

# print results
budget_included_proportion
```

```
## [1] 0.08870858
```

```r
budget_not_included_proportion
```

```
## [1] 0.9112914
```

```r
# Find the top 5 most expensive movies
top_5_expensive <- movies %>%
  filter(!is.na(budget)) %>%
  arrange(desc(budget)) %>%
  head(5)

# rint the top 5 expenive movies
top_5_expensive
```

```
## # A tibble: 5 x 24
##   title      year length budget rating votes    r1    r2    r3    r4    r5    r6
```

```
##   <chr>     <int>  <int>  <int>  <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Spider-M~  2004    127 2   e8    7.9 40256   4.5   4.5   4.5   4.5   4.5   4.5
## 2 Titanic    1997    194 2   e8    6.9 90195  14.5   4.5   4.5   4.5   4.5   4.5
## 3 Troy       2004    162 1.85e8    7.1 33979   4.5   4.5   4.5   4.5   4.5  14.5
## 4 Terminat~  2003    109 1.75e8    6.9 32111   4.5   4.5   4.5   4.5   4.5  14.5
## 5 Waterwor~  1995    176 1.75e8    5.4 19325   4.5   4.5   4.5  14.5  14.5  14.5
## # i 12 more variables: r7 <dbl>, r8 <dbl>, r9 <dbl>, r10 <dbl>, mpaa <chr>,
## #   Action <int>, Animation <int>, Comedy <int>, Drama <int>,
## #   Documentary <int>, Romance <int>, Short <int>
```

```r
movies_with_budget <- sum(!is.na(movies$budget)) / nrow(movies)

# print
movies_with_budget
```

```
## [1] 0.08870858
```

```r
#3

# find the top 5 longest movies
top_5_longest <- head(movies[order(-movies$length), ], 5)

# print
top_5_longest
```

```
## # A tibble: 5 x 24
##   title      year length budget rating votes    r1    r2    r3    r4    r5    r6
##   <chr>     <int>  <int>  <int>  <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Cure for~  1987   5220     NA    3.8    59  44.5   4.5   4.5   4.5   0     0
## 2 Longest ~  1970   2880     NA    6.4    15  44.5   0     0     0     0     0
## 3 Four Sta~  1967   1100     NA    3      12  24.5   0     4.5   0     0     0
## 4 Resan      1987    873     NA    5.5    12   0     0     4.5   0     0     0
## 5 Out 1      1971    773     NA    6.7    20   4.5   4.5   4.5   0     4.5  14.5
## # i 12 more variables: r7 <dbl>, r8 <dbl>, r9 <dbl>, r10 <dbl>, mpaa <chr>,
## #   Action <int>, Animation <int>, Comedy <int>, Drama <int>,
## #   Documentary <int>, Romance <int>, Short <int>
```

```r
#4

short_movies_duration <- movies$length[movies$short == 1]
```

```
## Warning: Unknown or uninitialised column: `short`.
```

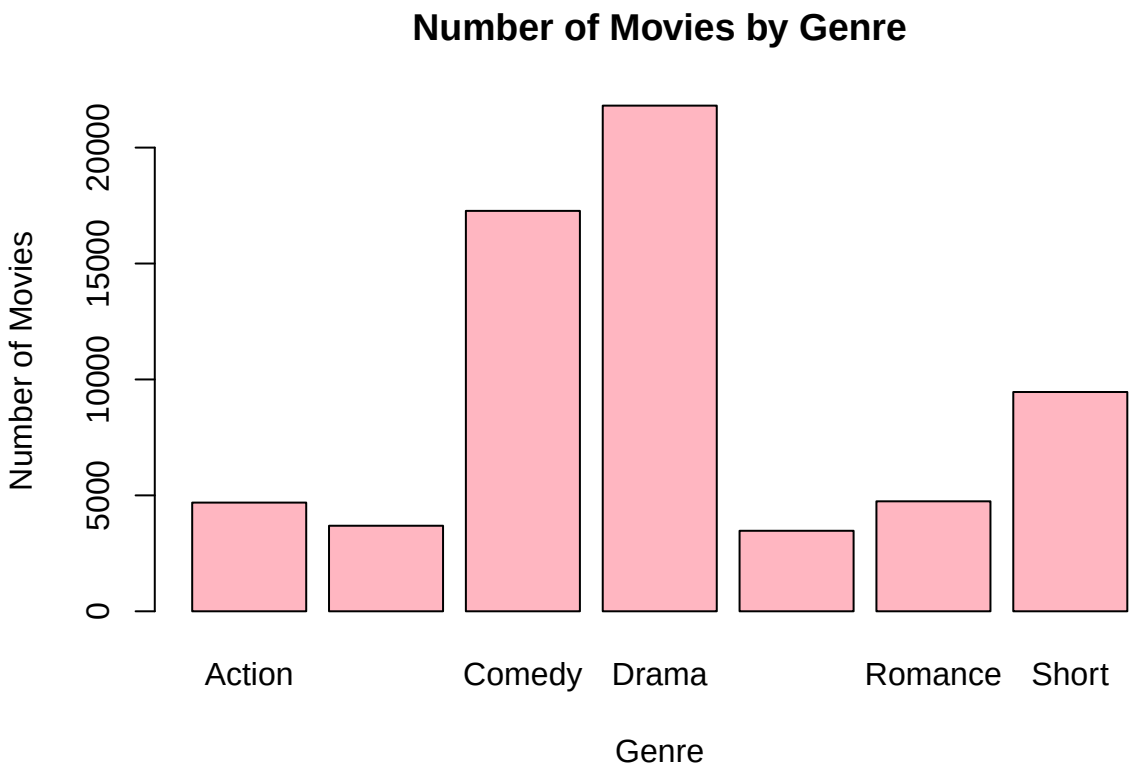
```r
if (length(short_movies_duration) > 0) {
  shortest_short <- min(short_movies_duration, na.rm = TRUE)
  longest_short <- max(short_movies_duration, na.rm = TRUE)
} else {
  # Handle the case where there are no short movies with duration information
  shortest_short <- NA
  longest_short <- NA
}

# Print
shortest_short
```

```
## [1] NA
```

```
longest_short
```

```
## [1] NA
```

```
#5
# Calculate the # of movies in each genre category
genre_counts <- movies %>%
  select(Action, Animation, Comedy, Drama, Documentary, Romance, Short) %>%
  colSums()

# plot the counts using the bar plot, make it pink!
barplot(genre_counts,
        main = "Number of Movies by Genre",
        xlab = "Genre",
        ylab = "Number of Movies",
        col = "lightpink")
```

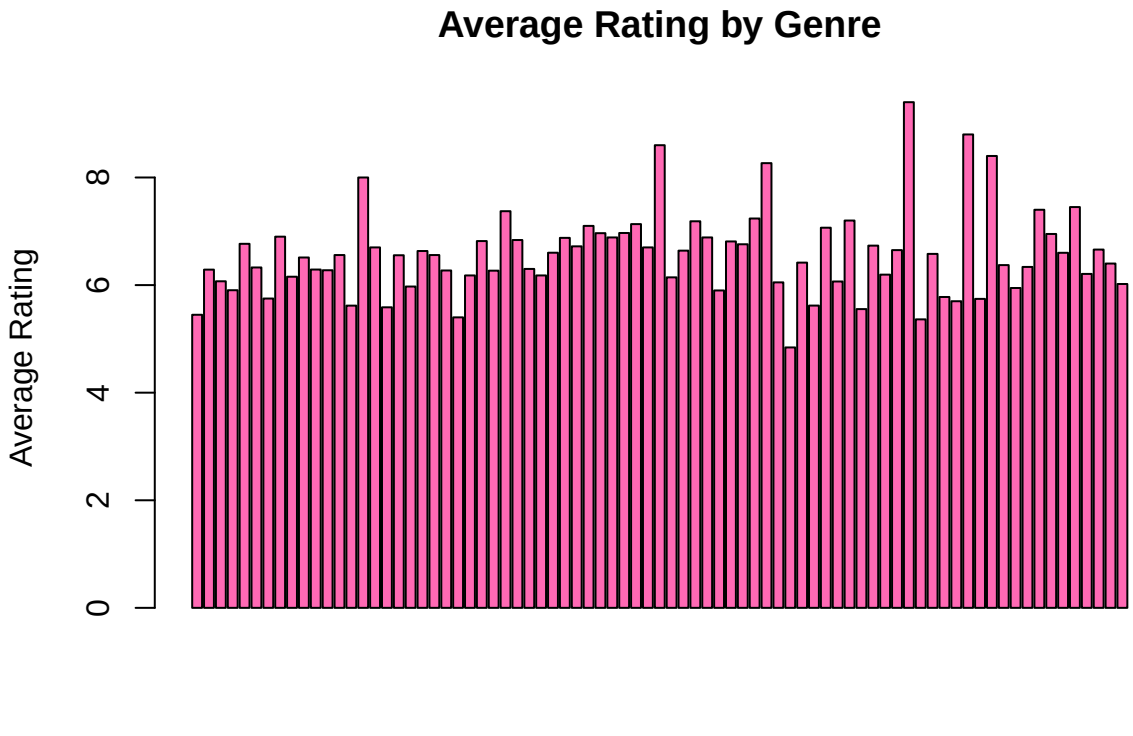



```
#6

# Calc the average rating for each genre
genre_averages <- movies %>%
  select(Action, Animation, Comedy, Drama, Documentary, Romance, Short, rating) %>%
  group_by(Action, Animation, Comedy, Drama, Documentary, Romance, Short) %>%
  summarize(average_rating = mean(rating))
```

```
## `summarise()` has grouped output by 'Action', 'Animation', 'Comedy', 'Drama',
## 'Documentary', 'Romance'. You can override using the `.groups` argument.
```

```
# lot the averages using a bar plot
barplot(genre_averages$average_rating,
        main = "Average Rating by Genre",
```

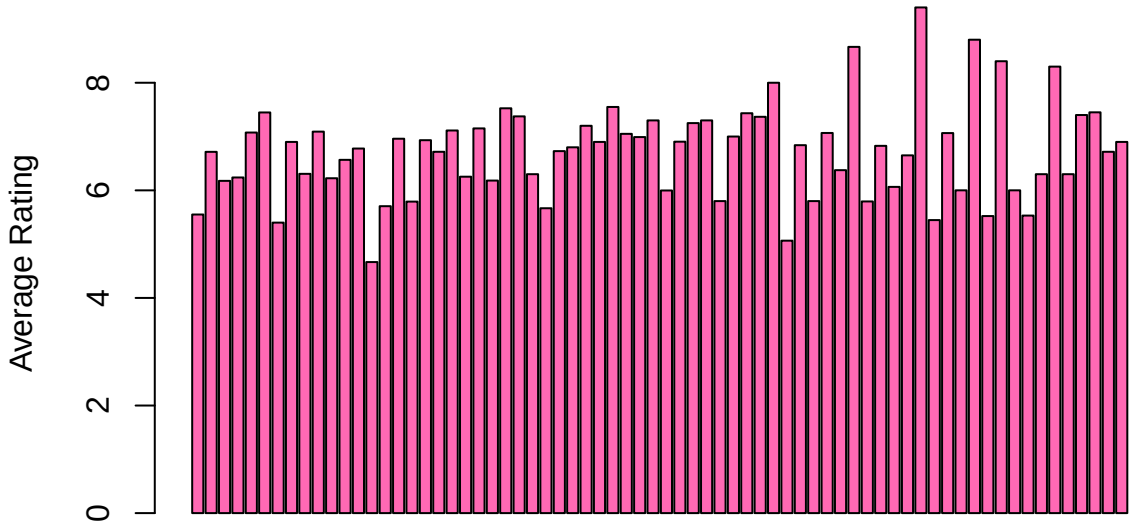
```
        xlab = "Genre",
        ylab = "Average Rating",
        col = "hotpink")
```




```
#7
# calc the average rating for each genre in the 2000-2005 period
genre_averages_2000_2005 <- movies %>%
  filter(year >= 2000 & year <= 2005) %>%
  select(Action, Animation, Comedy, Drama, Documentary, Romance, Short, rating) %>%
  group_by(Action, Animation, Comedy, Drama, Documentary, Romance, Short) %>%
  summarize(average_rating = mean(rating))
```

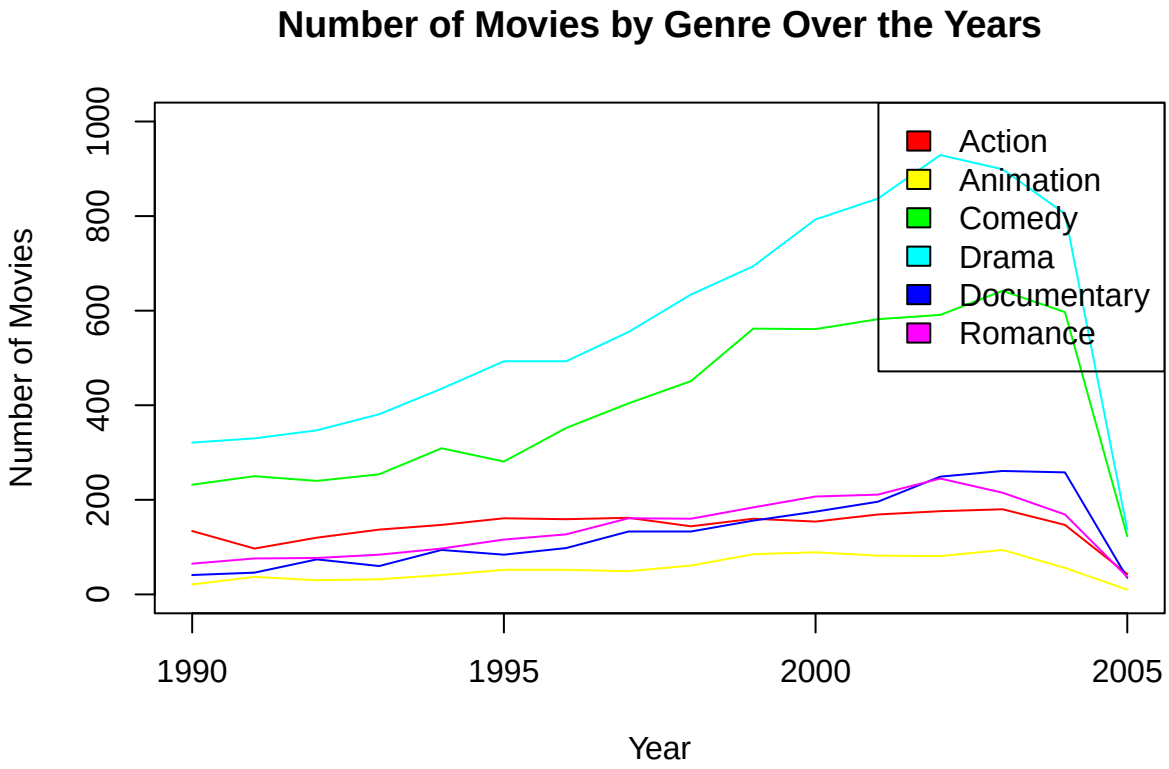
```
## `summarise()` has grouped output by 'Action', 'Animation', 'Comedy', 'Drama',
## 'Documentary', 'Romance'. You can override using the `.groups` argument.
```

```
# plot the averages using a bar plot, hot pink!
barplot(genre_averages_2000_2005$average_rating,
        main = "Average Rating by Genre (2000-2005)",
        xlab = "Genre",
        ylab = "Average Rating",
        col = "hotpink")
```

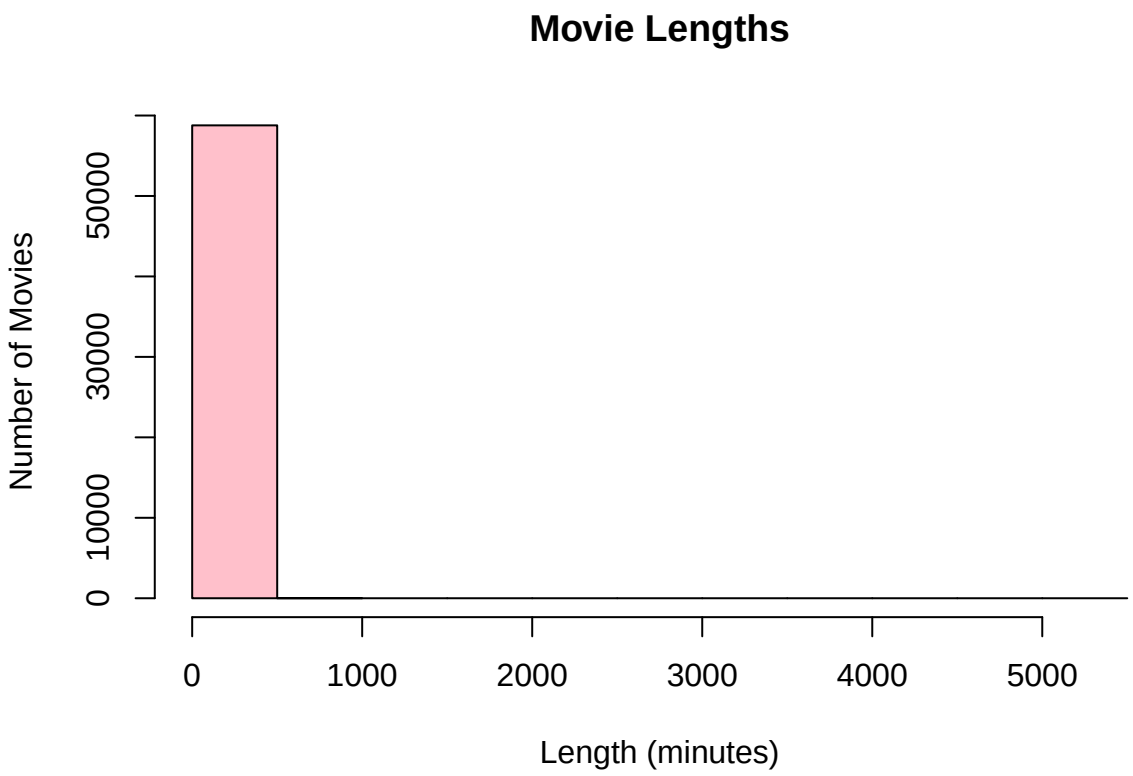



```r
#8
# filter movies by year (from 1990 to the last year recorded)
filtered_movies <- movies %>%
  filter(year >= 1990)

# select the fist 6 genres (excluding "Short")
genres_to_plot <- c("Action", "Animation", "Comedy", "Drama", "Documentary", "Romance")

#create an empty plot to initialize the figue
plot(1, type = "n", xlim = c(1990, max(filtered_movies$year, na.rm = TRUE)), ylim = c(0, 1000),
     main = "Number of Movies by Genre Over the Years",
     xlab = "Year",
     ylab = "Number of Movies")

#create a list of colors for the genres
genre_colors <- rainbow(length(genres_to_plot))

# Initialize a vector to store legends
legend_text <- character(length(genres_to_plot))

#loop through the genres and plot the number of movies by year
for (i in 1:length(genres_to_plot)) {
  genre <- genres_to_plot[i]
  genre_movies <- filtered_movies %>%
    filter(.data[[genre]] == 1) %>%
    group_by(year) %>%
    summarize(n = n())

  lines(genre_movies$year, genre_movies$n, type = "l", col = genre_colors[i])
  legend_text[i] <- genre
```

```
}

# Add a legend to the plot, more pink
legend("topright", legend = legend_text, fill = genre_colors)
```




```
#9
hist(movies$length,
     main = "Movie Lengths",
     xlab = "Length (minutes)",
     ylab = "Number of Movies",
     col = "pink")
```


Movie Lengths
Number of Movies
0
10000
30000
50000
0
1000
2000
3000
4000
5000
Length (minutes)